The PowerQUICC II MPC8260

Purpose:

- The General Architecture module offers a brief overview of the MPC8260, which was the first member of the PowerQUICC II microprocessor family.

Objectives:

- To help you understand the three basic components of the MPC8260 architecture, the Core Processor, the Communication Processor Module (CPM) and the System Interface Unit (SIU), as well as their main processing functionality.

Contents:

- Functionality of the Core, CPM, and SIU followed by an operating example.

Learning Time:

- There are 12 pages for this module and it will take approximately 17 minutes

The General Architecture module offers a brief overview of the MPC8260, which was the first member of the PowerQUICC II microprocessor family.

The objective is to help you understand the three basic components of the MPC8260 architecture, the Core Processor, the Communication Processor Module (CPM) and the System Interface Unit (SIU), as well as their main processing functionality.

The contents of this module contain functionality of the Core, CPM, and SIU followed by an operating example.

There are 12 pages for this module and it will take approximately 17 minutes.

Architecture

The MPC8260 consists of three major blocks:

- A 32-bit core derived from the PowerPC 603e, with MMU and cache
- A system interface unit to provide major interfacing logic to external devices
- A communications processor module consisting of a 32-bit RISC processor and a number of independent serial controllers and components to provide flexible, programmable solutions to communications systems

The PowerQUICC II consists of three major blocks. The core processor is a 32-bit CPU derived from the PowerPC 603e. This was based on the Power PC architecture, and complies with that programming model. It includes MMU and Cache providing improved performance over the earlier PowerQUICC family.

A CPU cannot operate usefully without the glue-logic to provide the interfacing and peripherals necessary to handle the housekeeping tasks required within an application. The serial interface unit, referred to later as the SIU provides these major needs.

Finally, within the communications environment that the MPC8260 is intended, the main protocol handling functions are provided by the communications processor module, commonly referred to as the CPM. This consists of another32-bit RISC processor specifically designed for handling all the functionality of the communications modules and associated functions. It operates on microcode, generally contained in an internal ROM, and offers a variety of selectable protocols and other operations. The result is a number of serial controllers and components providing flexible, programmable solutions to communications systems.

Features – CPU + General

- Dual-issue core version of PowerPC MPC603e
- 64-bit data and 32-bit address PowerPC bus (supports multi-master)
- Instructions and data cache - each 16kB physically addressed, 4-way set associative
- MMU for each of instruction and data access.
- Low power operation (<2.5W @ 133MHz, 2.5V internal and 3.3V i/o)
- Separate power supply for internal logic (2.5V) and I/O (3.3V)
- Common on-chip processor test interface
- Disable CPU mode

The basic features of the CPU and general package are shown here. The core is a dual-issue CPU version of the PowerPC MPC603e. This means that it can issue two instructions at the same time, and conforms to the Power PC architecture.

The data bus is 64-bits wide and address bus is 32-bits wide, and it supports multi-master operations.

The CPU has separate instruction cache and data cache, each being 16kB, physically addressed, and four way set associative.

Related to each cache is an MMU providing memory management for both program and data space. This is necessary to provide additional separation and protection for the memory above that offered by the Memory Controller, even if address conversion is not necessary.

The MPC8260 offers low power consumption, the original device requiring less than 2.5 Watts at 133 MHz.

To provide the best performance and interface options, the internal logic operates at 2.5 volts, while the input output logic operates at 3.3 volts. This enables standard 3.3 volt devices to interface with it but also requires the use of two separate power supplies.

For debugging purposes, a common on-chip processor test interface is provided.

As an option, the CPU can be disabled for situations where a more powerful external CPU may be required, or a multiple PowerQUICC II or multi-processor environment is necessary but all the CPU's are not required. This would include systems where more serial devices are required than this device offers, but only one CPU is sufficient.

Features – System Interface Unit

- Clock synthesizers with separate PLLs for core and CPM
- Reset control
- Real-time clock (counter ! - no battery backup)
- Periodic interrupt timer
- Hardware bus monitor and software watchdog
- 32-bit data and 18-bit address local bus
- 12-bank memory controller providing glueless interface to most memory types
- JTAG test access port

The system interface unit provides the glue logic to enable a working system. Among the most important functions available are the following: Clock synthesizers with separate phase-locked-loops for both the core and CPM. This provides the ability for both to operate at different frequencies. From the base frequency further options are available to select the required frequencies for sub-modules.

Reset control enables the automatic reset of the device under certain error conditions, as well as the selection of optional configurations following reset.

There are a number of counters available for the user, including one able to provide a seconds count given the appropriate input frequency. However, there is no protection for power failure or reset.

The periodic interrupt timer is capable of generating an interrupt based on matching a programmed value.

For protection from lock-up due to hardware and software failures there is a hardware bus monitor and a software watchdog.

Each offers the choice of either a reset or exception following a failure.

As an alternative to the standard 60x bus for transferring data there is also a local bus. This provides 32-bit addressing and an 18-bit data bus. One of the major concerns with a system of this complexity is a transfer bottleneck, where a device cannot access the bus during transfers by other devices. The local bus can relieve this problem by providing an alternative, parallel path, so that two transfers can take place concurrently.

For providing the necessary control signals between the MPC8260 and various memory devices and peripherals there is a 12 bank memory controller. It consists of 3 different types of programmable devices enabling glueless interfaces to most types of memory.

For testing the internal functionality there is a standard JTAG test access port.

Features – Communications Processor Module

- Embedded 32-bit RISC controller
- Interface to core and external system
- Serial DMA for all internal serial devices
- Pin controls for all programmable pin functions
- Virtual DMA for independent data transfers
- Three fast communication controllers supporting
  - 10/100Mbit Ethernet/IEEE 802.3 CSMA/CD via media independent I/F
  - ATM full duplex SAR @ 155 Mb/s via UTOPIA; and TDM interface
  - HDLC up to T3 rate
  - Transparent mode
- Four serial communications controllers (identical to MPC860)
- Two serial management controllers (identical to MPC860)

At the heart of the CPM is a 32-bit RISC processor that controls the individual modules. This will be referred to in the future as the CP. It is completely separate from the rest of the system and inaccessible to the user. It operates from microcode in an on-board ROM, but can also use downloaded code in dual-port RAM. All of the protocol handling for the serial channels, along with CPM functionality is controlled by this processor.

The CPM interfaces to both the core and external system, allowing transfer of data both internally and to external systems.

For fast transfers of data between serial controllers and memory there are DMA controllers for each channel to ensure the best possible performance.

Because there are more input and output functions available than pins to connect to, many pins are programmable. This means that if certain connections are required, particularly with respect to the serial channels, the pins must be programmed to suit. Controls are provided for this.

An additional aid to performance is a virtual DMA controller to support independent data transfers within the system, both internally and externally. This relieves the CPU from mundane data transfer tasks, once again improving performance.

The major improvement over earlier versions of communications processors is three new fast communications controllers.

These appear to be very similar to the serial communications controllers, but are completely redesigned to enhance performance. They offer HDLC performance up to T3/E3 rate; Up to 100 Mbps ethernet via a media independent interface;

ATM full duplex segmentation and reassembly functionality up to 155 Mbps per second via the UTOPIA interface.

The rate does depend on the adaptation layer used. Transparent mode is also available.

The CPM has four serial controllers identical to those on the MPC860.

The CPM has two serial management controllers identical to those on the MPC860.

Features – CPM (continued)

- One serial peripheral interface (identical to MPC860)
- One I^2C controller (identical to MPC860)
- Two serial interfaces supporting four time division multiplexers each which can support T1, CEPT, T3/E3, PCM, ISDN, IDL, GCI, and user-defined interfaces
- Two multi-channel hardware controllers, each supporting up to 128 full-duplex channels:
  - Full support for super and sub-channels
  - Capable of defining four subgroups of 32 channels
  - Subgroups can be flexibly multiplexed to TDMs
  - Independent transmit and receive routing and control
- Eight independent baud rate generators plus 20 input clocks for independent clocking of each serial controller

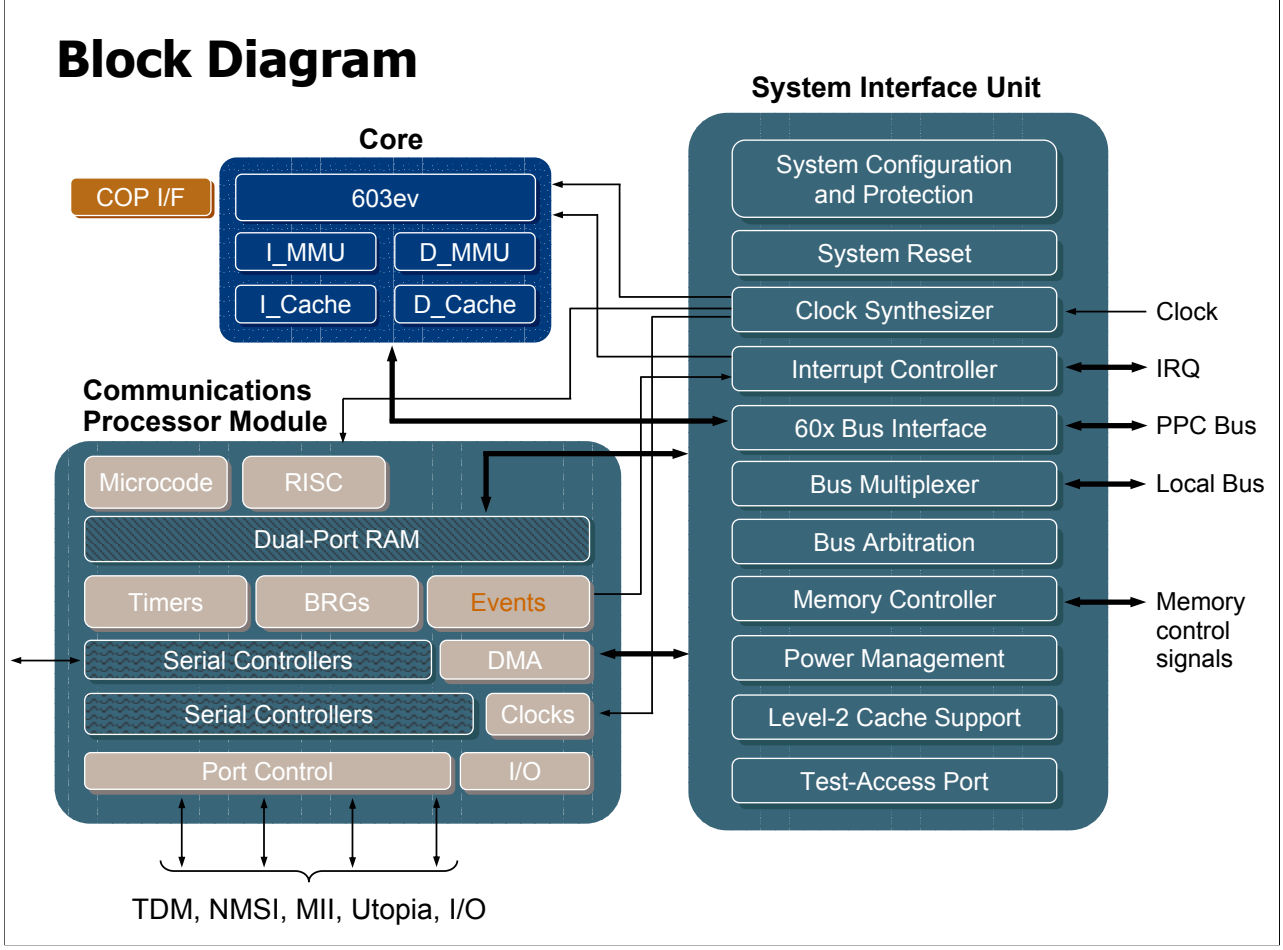
Additional features of the CPM include one serial peripheral interface identical to those on the MPC860.

It includes one I^2C controller also identical to the MPC860.

The CPM includes two serial interfaces four time division multiplexers each, a total of 8 TDM ‘s, supporting T1, CEPT, T3/E3, PCM, ISDN, IDL, GCI and user defined interfaces.

It has two multi-channel controllers , each supporting up to 128 full duplex channels, a total of 256 channels. Each controller supports super and sub-channeling, can define up to four subgroups, sub-groups can be flexibly multiplexed to TDM’s, and transmit routing and control can be independent from receive.

And finally, the CPM features 8 baud rate generators plus 20 clock inputs for independent clocking of serial channels, which provide ample possibilities for achieving the required rates



The Block diagram gives a good overall picture of the PowerQUICC II , showing the three major blocks, the functions within them, and some of the basic connections between them. You can see the relationship between the functions and their blocks.

The cop test interface is associated directly with the core, as are the MMU and cache. The MMU is shown between the cache and CPU indicating that the cache contains physical tags. This improves performance of cache to memory interaction.

Notice that the MMU and cache are located within the core and therefore are never involved with any transactions other than to the core.

The SIU shows the major functions, some of which were not outlined earlier. An external oscillator is required to drive the clock synthesizer and there are separate outputs from the clock synthesizer to the core and CPM. System Configuration and protection allows flexible configuration of the system start-up and how the device deals with fatal errors.

The interrupt controller provides programmable control of how the multitude of events generated by the CPM are reported to the core, and how they relate to other external and SIU interrupts. Internally the interrupt controller generates a request to the core, but can also generate a request to an external master.

The 6Ox, bus interface, bus multiplexer, and bus arbitration are closely related. Many transactions can be routed via either the 6Ox, bus or local bus, and all transactions on the bus must arbitrate for access. Clearly, with all the functions available, all of which at some time will need to transfer data, as well as control information, multiple requests will regularly be made to the bus. These must be prioritized and controlled properly.

Programmable priority selection is provided to allow the user to define the required relationship. A separate input is used for PLL power and VCC syn, to achieve good stability. Full Power mode means both the main PLL and Core PLL operate. Stop mode means that the main PLL is operating while the core PLL is stopped.

For a description of the PLL power management, refer to “Clocks and Power Control”, subsection “Basic Power Structure”, in the user’s manual.

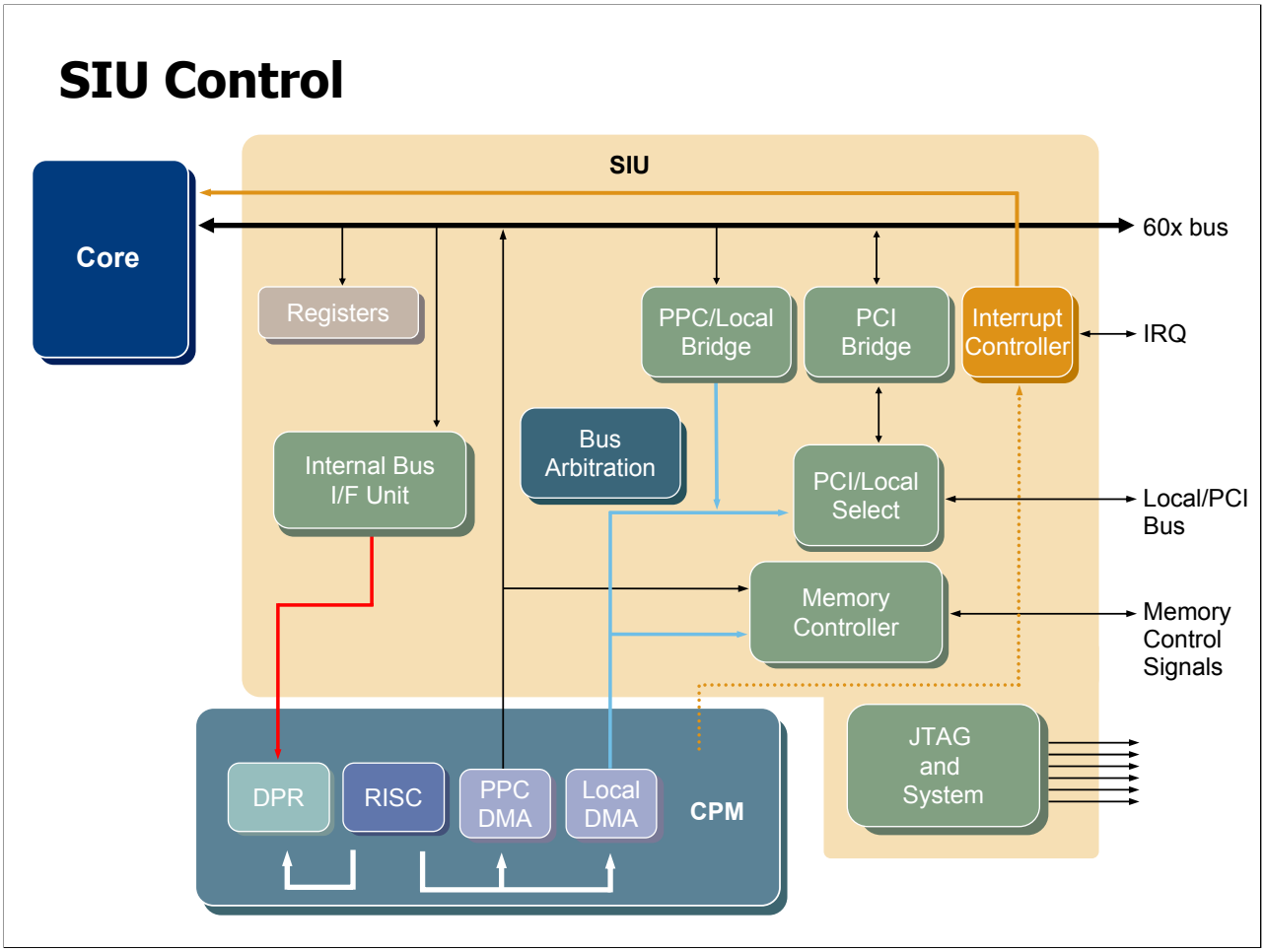
For the rest of the course the RISC controller will be designated the CP. The CPM diagram indicates that the CP and ROM microcode are separated from the peripherals by the dual-port RAM. This is to indicate that the dual-port RAM is the interface between the controller and the peripherals. The user provides the information and parameters to select the required control features by writing to the dual-port RAM, using the core or external master. The CP will access the dual-port RAM as needed to determine requirements, and operate on them accordingly.

Along with the serial controllers are a number of hardware and software timers which can be used independently as required;

Input output controls enabling independent pin functions; and port controls allowing the independent selection of multiple pin functions. Clearly the pin function selection is vital to the use of the function, because we have many toys to play with , but we can’t have them all at the same time. Notice that recognized events are reported to the interrupt controller in the SIU.

The serial interfaces available to the system, which are entirely dependent on the port controller, are time division multiplexed, non-multiplexed serial interface, media independent interface, UTOPIA, and individual input output pins.

SIU Control



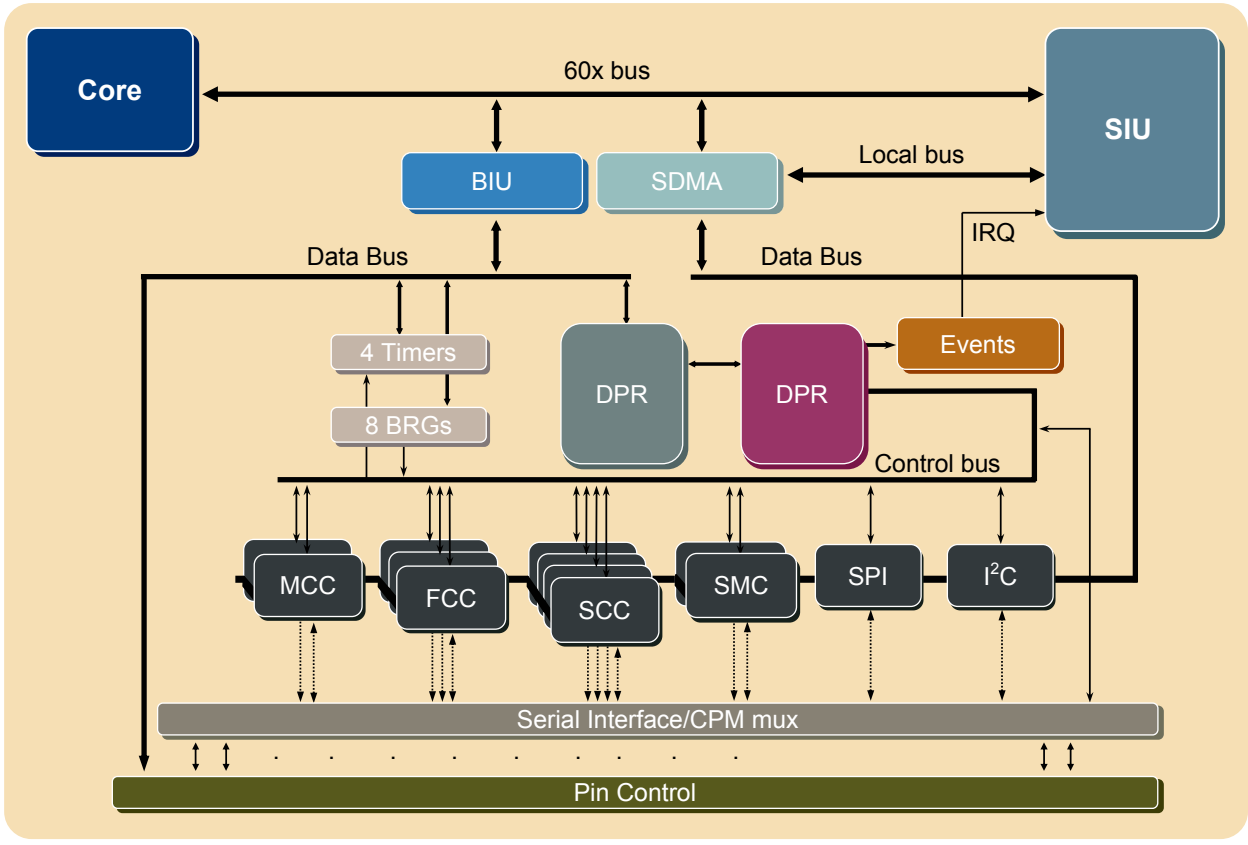
The SIU provides the general glue logic for the system providing the bus interface between the system and CPM, bridging between the 60x and local bus, the memory controller, system configuration and protection, JTAG, interrupt control and bus arbitration.

There are two buses, the 60x bus for system use, particularly with multi-processor systems, and Local bus which can be used to reduce bus latency for internal peripherals. The Local bus becomes the PCI bus when implemented on later family members. The COP interface replaces the background debug mode of the earlier devices, providing debug and control of the system including flash programming.

Direction of arrows indicates who is the controller, not data flow. For example, the core can initiate reads or writes via dual-port RAM but DPR never initiates transfers. Registers are accessed from the 60x bus. Transactions on the 6Ox bus can initiate transfers to the local bus but not vice versa. DMA can initiate transfers to both the 60x bus and Local bus.

The memory controller is initiated by transactions on the 60x bus and local bus which are requested by the core, external master, or the DMA controller. The CP initiates accesses to dual port RAM and the DMA controller. CPM events write to the interrupt controller. Effectively the core, internal logic, and the external system are linked on the 60x bus. All transactions involve bus arbitration.

CPM Block Diagram



The block diagram shows the major components of the CPM. The serial controllers transfer serial data through the serial interface selected and the pin control logic at one end, and transfer parallel data via the 60x bus or local bus at the other. The controllers operate the required protocol, controlled by the RISC, which obtains the necessary parameters from DPR and operates as defined by code from it's internal ROM. Any status information is returned to DPR by the RISC. It also indicates events via registers specifically for the purpose, that can report to the interrupt controller.

There are different serial controllers handling a range of protocols and performance options to provide a great deal of flexibility for a system. It should also be noted that not all of the options are available at the same time, and careful consideration must be made of both the connection possibilities, and performance issues.

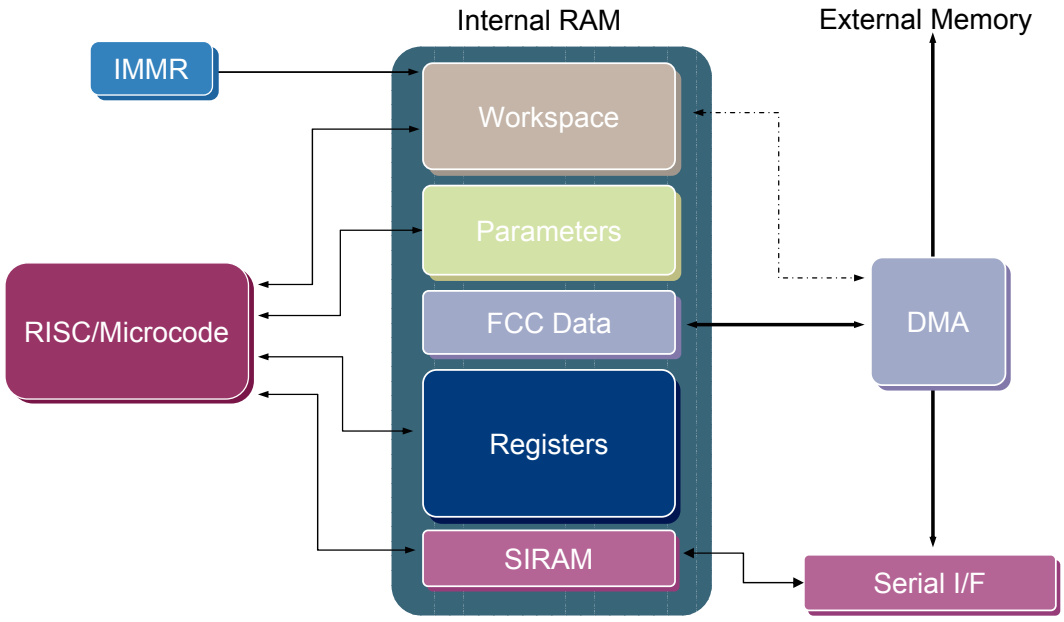
The I^2C and SPI controllers provide simple transfers of information between local system peripherals. The SMCs provide basic uart and transparent connections, and GCI support. The SCCs offer the largest range of protocol options from HDLC and transparent mode which is more comprehensive than that of the SMCs, HDLC to 10 Mbps ethernet. FCCs are higher performance versions of the SCCs offering HDLC fast ethernet to 100 Mbps, and ATM to 155 Mbps. The MCCs provide multi-channel support in transparent or HDLC mode.

The baud-rate generators can provide the clocks to the serial controllers, which can also be clocked from external pins, and the timers could be used independently, or in conjunction with the controllers for applications purposes. Data can also be transferred directly through individual pins. How the serial controllers transfer data is controlled by the serial interface, and the connections to the external world must be defined by the pin control logic.

The CPM is accessed via the bus interface unit to initialize peripherals and internal dual-port RAM, using either the core or an external master. There is no direct access to the RISC controller, only through DPR as the interface. RISC microcode controls peripherals using information in dual-port RAM. Data flow between peripherals and memory is controlled using independent serial

DMA which can independently transfer via the 60x bus and local bus. As events occur, which can be masked, they issue a request to the SIU interrupt controller, which can provide an interrupt request to either the CPU or an external master.

Internal RAM



Reference Manual 13-15

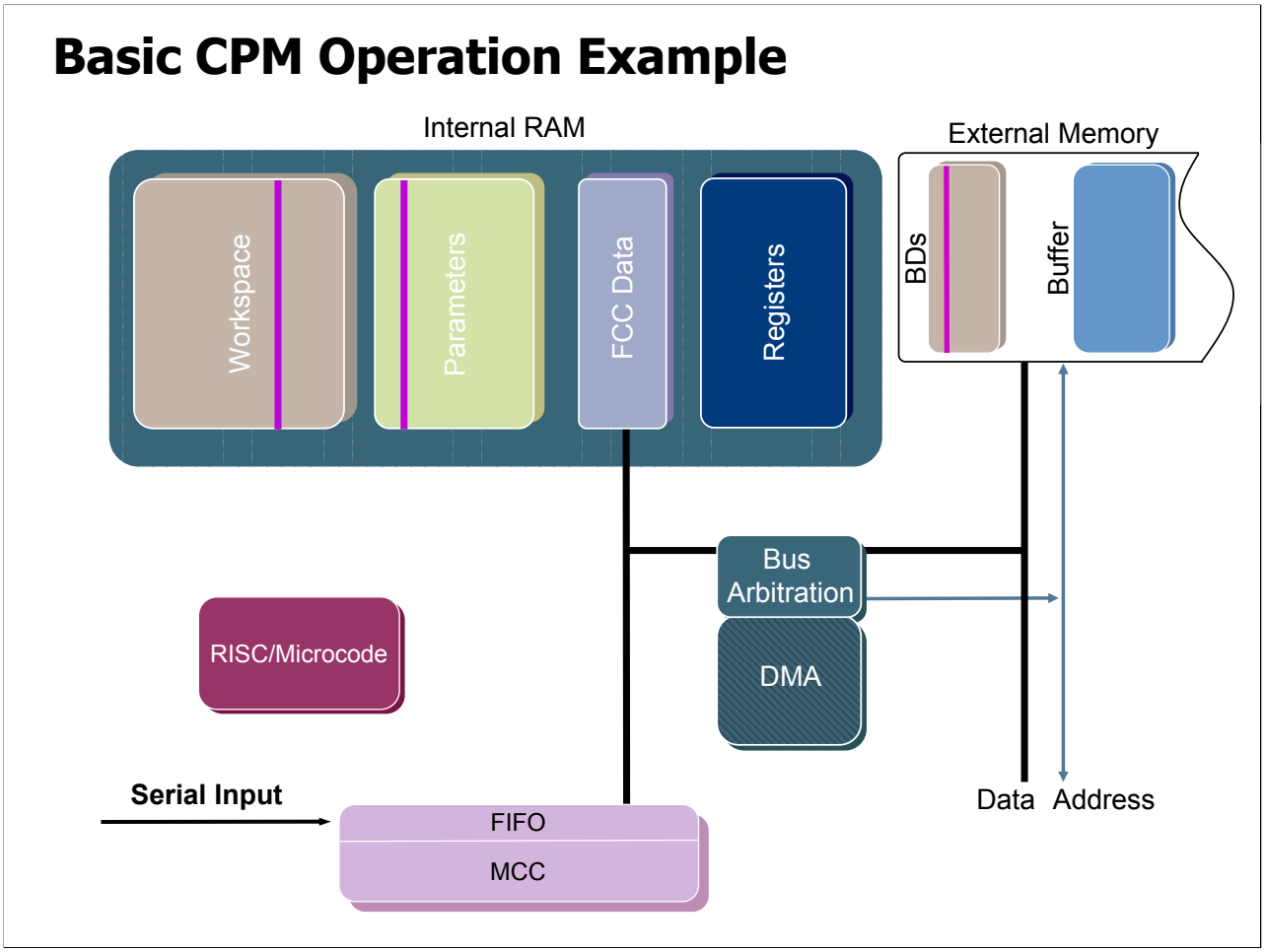
Internal peripherals are controlled by the embedded RISC controller referred to as the CP which stands for communications processor. This performs instructions held within it's own ROM that are selected by the function defined in registers and parameters in internal dual-port RAM, which is the interface between the CP and the rest of the world.

The internal RAM is defined by five basic sections which have flexible allocation: Workspace is where the application can place run time variable such as buffer descriptors and data. Workspace also is used for loading microcode; The parameters can consist of the fixed variables for the operation of the risk related to the peripherals; The FCC manages DMA transfers, providing a buffer of 32 bytes per channel to improve throughout for high speed data; The Registers are in fact latches which have pre-defined functions related to peripherals, to provide the selectability necessary. They can be accessed like memory locations with the exception of event registers, which are used for the CP to report events to the interrupt controller.

SI RAM is used to program the serial interfaces for time division multiplexed functions. It defines the time slot and controller related to it.

IMMR is an internal SIU register which defines the location of internal RAM on any 128 kilobyte boundary within the system memory map, and can be configured, to a limited extent, at hard reset. This must be defined at start-up to ensure that DPR is accessible once the processor is ready for initialization.

Basic CPM Operation Example



This slide is intended to provide a general idea of how the CPM controls data transfers. In this case it indicates the basic operations involved in transferring data from the input of an MCC to memory. Similar principals are involved in all serial controller operations. A fundamental principal is that a considerable number of steps occur in the process.

Initially serial data received is transferred into the controller FIFO. Before the FIFO is full a signal is sent to the CP indicating that data is available and must be transferred out of the FIFO before it is full. This effectively generates an interrupt to the CP to perform that specific task, which is one of many it could be performing. The CP operates to a prioritized schedule, and when it reacts to this interrupt it transfers the data in the FIFO, in parallel form into dual-port-ram using the DMA engine. This now provides more room in the FIFO for more incoming data.

The CP then reads parameters in dual-port-ram to find the location of pointers in workspace that provide the address of buffer descriptors in external memory to the DMA engine.

Now the CP needs to access the buffer descriptors, and so bus arbitration takes place, and when the bus is free the buffer descriptor address is driven onto the address bus. This gives access to the buffer descriptor, which provides necessary information about the buffers.
The CP reads the buffer descriptor from which it determines if the buffer is ready and where it is located, and assuming the buffer is ready again arbitrates for the bus.

When the bus is free the DMA engine transfers the data from dual-port-ram to the buffer, which completes the process of getting incoming serial data from the FIFO to memory. However, there is unlikely to be only one transfer needed, since serial data could be streaming in far more than a single FIFO quantity. This process continues as long as data is being received. A similar process occurs for transmitted data. Consider this—What would happen if the CP did not recognize the FIFO request, or was unable to handle it, before it was full? Or, bus arbitration withheld access to the bus for too long? Or, a buffer is not ready? If the FIFO data is not read before it is full, then data will be overwritten and therefore lost. If bus arbitration withholds access to the bus then FCC data in dual-port-ram will be overwritten, again resulting in data loss. The same applies if the buffer is not ready.

All of this must be considered, and will affect the possible performance of the controller. Another consideration is the schedule of the CP. If it is too busy to attend to this process because of all the other functions it deals with, then again performance is affected. The serial input data rate is also a consideration. Of course, the serial data rate is considerably slower than the parallel transfer rate, so the problem is not as dire as it could be, but nevertheless all of these details must be factored into the performance calculations.

Parameter RAM allocation

Page	Address	Peripheral	Size
1-4	8000	SCC1-4	4x256
5-7	8400	FCC1-3	3x256
8	8700	MCC1	128
	87FC	SMC1 base	2
	87FE	IDMA1base	2
9	8800	MCC2	128
	88FC	SMC2 base	2
	88FE	IDMA2 base	2
10	89FC	SPI base	2
	89FE	IDMA3 base	2
11	8AE0	Timers	16
	8AF0	Rev_num	2
	8AF8	Rand	4
	8AFC	I^2c base	2
	8AFE	IDMA4 base	2

Reference Manual 13-18

This diagram indicates parameter RAM allocation. Parameter RAM is defined as 21 blocks of different sizes, all relating to specific functions. When the CP needs to handle a peripheral it will access the parameters for it. It must know exactly where to look for the information, and so these parameters are at pre-defined locations which are defined as offsets from the top of dual-port RAM.

The first four blocks are of 256 bytes each used for each of the SCC‘s. Three similar blocks are used for the FCC‘s.

You can examine the list to see what peripherals have parameters, how large they are in bytes, and the address.

Remember that what is defined as address is actually an offset from the top of dual-port RAM, which could be located almost anywhere in the overall memory map. The IMMR, which stands for internal memory map register, defines the top of dual-port RAM, and so the exact location of the required parameter is a combination of the IMMR and its offset. The exact make up of the parameters can be seen in the manual section related to the specific peripheral and will be covered in the modules of the related sections of this training.

The left column of the table indicates a page number. This is used when initializing parameters when commands have to be issued to the CP using the command register. The appropriate page number is used to indicate the parameters related to the command. An important feature to be aware of is that whenever parameters are initialized, a command must be issued to the CP.

That concludes the Architecture Module. For more information on the general PowerQUICC II Microprocessor architecture, please reference the Overview section of the Users Manual.